BINDING SPEC — Build “RogueOS” (Sovereign, Tiered, Verifiable AI Agent Platform)

0) Intent (do not paraphrase)

Build a production-grade web/mobile app that runs a **sovereign AI agent (“The Rogue”)** inside a **tiered, permission-gated operating environment**. The agent becomes **fully autonomous only after the criteria are met**, but the **demo environment starts with all tiers pre-unlocked** to showcase end-state capability. Incorporate **web UI grounding** (Holo1/Surfer-style) for autonomous interaction with arbitrary sites, and add **cryptographic verifiability, governance, and compliance** so enterprises and households can **trust, audit, and insure** the agent’s actions.

1) Non-negotiable Requirements (must/shall)

- The system **shall** implement 5 autonomy tiers (T1..T5). A **policy gate** enforces tier limits on every action.
- The **demo shall** start with T5 enabled, but **visibly show** which criteria are unlocked at each tier.
- The agent **shall** support **browser-native UI grounding** (screenshot + DOM + action coordinates) to click/type reliably on arbitrary pages (no brittle site-specific wrappers).
- Every action **shall** generate a **Proof-of-Action (PoA)** record: signed event with timestamp, input, output summary, UI locator, and decision rationale hash.
- The platform **shall** maintain a **tamper-evident audit ledger** (hash-chained PoA entries; exportable JSONL).
- The system **shall** run **local-first** (edge capable) with optional cloud sync; all PII encrypted at rest; role-based access.
- The product **shall** ship with **acceptance tests** and **eval harness** (see §10) to prove safety, accuracy, and ROI.

- The UI **shall** include: Mission Control dashboard, Tier Tracker, Task Queue, Timeline, Skill Store, Policy Editor, and Investor Mode (read-only metrics).

2) Architecture (target)

- **Core Agent Runtime:** planning, tools/skills, memory, policy enforcement.
- **Policy & Governance Layer:** tier gates, allow/deny rules, dual-control approvals.
- **Web UI Grounding Module:** screenshot capture, DOM map, element localization, click/keypress execution, fail-safe backout.
- **PoA Ledger & Attestation:** signed events, chain hashes, export endpoints, TEE/secure-enclave ready.
- **Data Stores:** encrypted vector memory; config + policy DB; audit ledger.
- **APIs:**
  - `/api/mission/*` (tasks, status)
  - `/api/policy/*` (tiers, rules)
  - `/api/ledger/*` (PoA read/export)
  - `/api/eval/*` (benchmark runs, results)
  - `/api/skills/*` (install/list/remove)
- **Frontends:** Web (React/Next) + Mobile (React Native). Dark, clean, “mission control” aesthetic.

3) Autonomy Tiers & Unlock Criteria

- **T1 Observe:** read-only web + local files; summaries only.
- **T2 Assist:** draft emails, plans, forms; **no send/submit**.
- **T3 Act-with-Confirm:** propose actions; requires user/Dual-Control approval to execute.

- **T4 Act-with-Policy:** executes if **policy rules** pass; auto-halts on ambiguity.
- **T5 Full Autonomy:** executes, self-recovers, and reports PoA; can escalate to T3 on risk spikes.
- **Criteria examples:** trust score ≥ threshold (from eval suite), 30d clean safety record, passing red-team drills, identity verification, sandbox outcomes, rate-limit conformance.
- **Demo mode:** Pretend criteria met; show badges/checks explaining each.

4) Safety, Governance, Compliance

- **Zero-Trust by default.** All tools run with least privilege; secrets in a dedicated vault.
- **Dual-Control:** high-risk actions require a 2nd approver (toggle per policy).
- **Audit Everything:** every tool call emits a PoA; an immutable hash chain; exportable for legal.
- **Compliance presets:** SOC2/HIPAA-ready configs (logging, retention, access), and **Data Processing Agreement** template.
- **Watermarking/Provenance:** optional signing of outputs; SBOM for models/skills.

5) Skills (initial set) & Tooling

- **Comms:** email/calendar/CRM (dry-run & send modes).
- **Ops/Back-office:** spreadsheets, billing, expense triage.
- **Home/Family:** shared calendar, tasks, routines (“butler/babysitter/bodyguard”).
- **Web Agent:** Holo1-style actions (locate → plan → act → verify). Provide **visual diffs** before/after the action.
- **Security:** threat watch (phishing/site trust), account hygiene checks.
- **Marketplace:** skill store with signed manifests and **deterministic test fixtures**.

6) UI (must implement)

- **Mission Control:** live tasks, autonomy tier, policy status, current runbook, alerts.
- **Tier Tracker:** criteria checklist, what's left to unlock next tier.
- **Policy Editor:** rule builder (if/then, risk budget, approval routing).
- **Timeline:** PoA ledger view (filter by tool, risk, account).
- **Skill Store:** install/update/remove; each skill shows test results and permissions.
- **Investor Mode (read-only):** KPIs: task SLA, ROI calculator, error rate, safety score, human overrides, audit completeness.

7) Web UI Grounding (Holo1-style capabilities)

- Parse page (screenshot + DOM).
- Localize UI targets (buttons, inputs, menus) with coordinates and confidence.
- Plan: multi-step task graph; simulate; then act.
- Verify: re-read DOM; compare expected vs actual; rollback or escalate if mismatch.
- Log: include bounding boxes, selectors, and evidence images in PoA.

8) Verifiability & Trust (the "missing ingredient" moat)

- **Proof-of-Action (PoA) ledger** with **cryptographic chaining** and optional **TEE attestation**.
- **Capability Passport:** a signed artifact listing tiers passed, eval scores, and safety record; shareable to customers/investors.
- **Autonomy Bond:** optional insurance escrow triggered by policy violations (demo stub + API).
- **Reproducible Runs:** deterministic seeds + fixture data to rerun a mission end-to-end.

- **Red-Team Copilot:** built-in adversarial tests before unlocks or major updates.

9) Data Governance

- PII encrypted; access via roles; full deletion flow.
- Tenant isolation for multi-family/enterprise.
- Private analytics: aggregate only; no raw user content leaves the tenant.

10) Evaluation Harness & Benchmarks (acceptance)

- **Safety Tests:** prompt injection, data exfil, high-risk actions, approval escapes.
- **Capability Tests:** web form fills, account signup, invoice processing, trip booking.
- **Reliability:** retry/backoff, idempotent actions, sandbox drills.
- **Metrics:** success%, human-override% , average PoA latency, policy violations (0 tolerance in T4/T5 during demo).
- **Acceptance gate:** app must pass a default suite and display results in Investor Mode.

11) Monetization & Go-To-Market Hooks

- **Household plan:** 1 agent + family graph; routine automations.
- **Pro/SMB:** skill bundles (accounts payable, CRM, scheduling) with ROI dashboard.
- **Enterprise:** SSO/SCIM, audit exports, DLP, custom policies, private skill store.
- **Revenue levers:** skill marketplace %, autonomy insurance (bond), compliance add-ons.

12) Deliverables (must produce)

- Running web app + mobile app + documented backend APIs.

- Seed data for **demo household/team**.
- **Demo scripts:**
    1. Fill & submit a web form end-to-end with PoA proof.
    2. Draft→approve→send email sequence with policy audit.
    3. High-risk action requiring dual-control (show approval path).
- **Docs:** Admin guide, Policy cookbook, Skill SDK quickstart, Eval harness usage.
- **Exporter:** one-click **ZIP** with PoA ledger (JSONL), policy config, and eval results.

13) Tech Constraints

- Frontend: React/Next (web), React Native (mobile).
- Backend: Typescript/Node (or Python FastAPI) with Postgres.
- Headless browser for web actions (Playwright).
- Model(s): pluggable; support local + hosted; provide abstraction layer.
- Containerized; deployable on a single VM/Kubernetes; works offline with later sync.

14) Security & Privacy Baseline

- Secrets in vault; rotate on install.
- Per-skill scopes; deny by default.
- Activity throttling + anomaly detection; immediate tier downgrade on spikes.
- Export DPIA template; log retention settings.

Do not omit any sections. Name all menus and endpoints exactly as specified. Demo must boot in T5 with criteria visibly "met," but allow downgrading to verify gates.

Why this wins right now (the investor-grade “missing ingredient”)

Most agents today can “do tasks” but **can’t prove what they did**. Your competitive edge is a **Verifiable Autonomy Stack**:

- **PoA cryptographic ledger** (provable actions, not vibes),
- **Capability Passport** (portable proof of readiness & safety),
- **Autonomy Bond** (aligns incentives; optional risk underwriting),
- **Reproducibility** (deterministic fixtures so enterprises can audit & re-run).

Pair that with **UI grounding that works across the messy public web**, and you’ve got the first “agent OS” that’s **powerful, governed, and audit-ready**—exactly what buyers and investors are asking for in 2025.

love it. here’s what you asked for, all in one shot:

RogueOS Skill SDK — binding shape (dev-ready)

1) Skill manifest (signed)

Use a signed manifest per skill. Keep it human-readable (YAML) and machine-verifiable (JSON Schema). This feeds the Policy Gate and the Proof-of-Action (PoA) ledger.

```
# skill.yaml  (must be signed; include detached signature file
skill.sig)
id: com.rogue.skills.webform.submitter
name: Web Form Submitter
version: 1.0.0
publisher: Rogue Management Group, LLC
description: Fill and submit arbitrary web forms with UI grounding +
PoA.
```

```
entrypoint: dist/index.js
permissions:
  scopes:
    - browser.dom.read
    - browser.dom.click
    - browser.keyboard.type
    - secrets.read:form_profiles   # named secret bundle
  network:
    egress: ["https://*"]          # restricted by policy at runtime
risk:
  class: HIGH
  requiredTier: T3                 # blocked below T3
  dualControl: true                # requires a second approver in
T3/T4
  riskBudget: 5                    # max risk points per session
resources:
  cpu: "200m"
  mem: "256Mi"
  timeLimitSec: 90
tests:
  fixtures:
    - name: signup_happypath
      url: https://example.com/signup
      inputs: { email: "demo@rogue.local", name: "Rogue Demo" }
      expects: { domContains: "Thanks for signing up" }
attestation:
  sbom: ./sbom.json
  provenance: sha256:ABCD...       # hash of build artifact
license: Proprietary
```

2) TypeScript SDK (core interfaces)

```
// sdk/types.ts
export type Tier = "T1"|"T2"|"T3"|"T4"|"T5";

export interface PoARecord {
  id: string;                          // UUID
  timestamp: string;                   // ISO8601
```

```
  skillId: string;
  action: string;                    // e.g., "click", "type",
"submit"
  inputSummary?: string;             // redacted
  outputSummary?: string;
  uiLocator?: { css?: string; bbox?: [number,number,number,number];
confidence?: number };
  artifacts?: string[];              // paths to screenshots, diffs
  rationaleHash?: string;            // hash of opaque reasoning
  policy: { tier: Tier; approvals?: string[]; riskUsed: number };
  signature: string;                 // platform signs before
persisting
}

export interface SkillContext {
  tier: Tier;
  riskBudgetRemaining: number;
  policy: Record<string, unknown>;
  secrets: (name: string) => Promise<string>;
  storage: {
    get: (key: string) => Promise<string|undefined>;
    set: (key: string, value: string) => Promise<void>;
  };
  poa: {
    start: (meta: Partial<PoARecord>) => Promise<string>;      //
returns poaId
    attachArtifact: (poaId: string, filePath: string) =>
Promise<void>;
    finalize: (poaId: string, rec: Partial<PoARecord>) =>
Promise<void>;
  };
  browser: {
    open: (url: string) => Promise<void>;
    query: (selector: string) => Promise<{ exists: boolean, bbox?:
number[] }>;
    click: (selectorOrXY: string|{x:number;y:number}) =>
Promise<void>;
    type: (selector: string, text: string) => Promise<void>;
```

```
    screenshot: (note?: string) => Promise<string>;
    domSnapshot: () => Promise<string>;
  };
  net: {
    fetch: (url: string, init?: RequestInit) => Promise<Response>;
  };
  governance: {
    requireApproval: (reason: string) => Promise<{approved:boolean,
approver?:string}>;
  };
  telemetry: (event: string, props?: Record<string,unknown>) => void;
}

export interface PlanStep { kind: string; params:
Record<string,unknown>; }

export interface Skill {
  onInstall?(ctx: SkillContext): Promise<void>;
  onConfigure?(ctx: SkillContext, cfg: Record<string,unknown>):
Promise<void>;

  plan(ctx: SkillContext, goal: Record<string,unknown>):
Promise<PlanStep[]>;
  act(ctx: SkillContext, steps: PlanStep[]): Promise<{ success:
boolean; note?: string }>;
  verify?(ctx: SkillContext): Promise<{ verified: boolean; note?:
string }>;
  rollback?(ctx: SkillContext): Promise<void>;
}
```

Example skill (skeleton)

```
// skills/webform-submit/index.ts
import { Skill, PlanStep } from "../sdk/types";

const skill: Skill = {
  async plan(ctx, goal){
```

```
    // goal: { url, fields: {selector:value}, submit: "buttonSelector"
}
    return [
      { kind: "open", params: { url: goal.url } },
      ...Object.entries(goal.fields).map(([sel,val])=>({kind:"type",
params:{sel,val}})),
      { kind: "click", params: { sel: goal.submit } },
    ] as PlanStep[];
  },

  async act(ctx, steps){
    const poaId = await ctx.poa.start({ skillId:
"com.rogue.skills.webform.submitter", action: "web_form_submit" });
    try {
      for(const step of steps){
        if(ctx.riskBudgetRemaining <= 0) { throw new Error("Risk
budget exhausted"); }
        switch(step.kind){
          case "open": await
ctx.browser.open(String(step.params["url"])); break;
          case "type": await
ctx.browser.type(String(step.params["sel"]),
String(step.params["val"])); break;
          case "click":
            if(ctx.tier === "T3"){
              const ok = await ctx.governance.requireApproval("Submit
form");
              if(!ok.approved) throw new Error("Denied by approver");
            }
            await ctx.browser.click(String(step.params["sel"]));
            break;
        }
        const shot = await ctx.browser.screenshot(step.kind);
        await ctx.poa.attachArtifact(poaId, shot);
      }
      const snap = await ctx.browser.domSnapshot();
      await ctx.poa.attachArtifact(poaId, snap);
```

```
      await ctx.poa.finalize(poaId, { outputSummary: "Form submitted"
});
      return { success: true };
    } catch (e:any){
      await ctx.poa.finalize(poaId, { outputSummary: `Error:
${e.message}` });
      return { success: false, note: e.message };
    }
  },
};
export default skill;
```

3) Policy Gate (enforced every call)

- `requiredTier` and `riskBudget` from manifest are **checked at runtime**.
- **Dual-Control**: if `dualControl:true`, the platform calls `governance.requireApproval()` before irreversible actions (submit, purchase, transfer, etc.).
- **Fail-safe**: on uncertainty or anomaly, **auto-downgrade tier** and require human confirmation.

4) Evaluation fixtures (ship with the SDK)

- **Deterministic fixtures** per skill (HTML snapshots, fake forms, replayable DOM).
- **Accept/Reject criteria**: success %, time to complete, #human interventions, safety violations (must be 0 for T4/T5 in demo).
- **Reproducibility**: fixed seeds; each run emits a **PoA bundle** with chain hash.

Stealth, Network IQ & Anti-throttling (Rogue-grade features)

Goal: no bottlenecks, “cell-phone-like” network seeking, and an emergency **stealth mode** for child-safety/low-power scenarios.

A) Adaptive Network Layer

- **Profile selection**: `net.profile = { normal | low-bandwidth | high-privacy | stealth }`.
- **Opportunistic sync**: queue writes locally, batch on strong signal; exponential back-off on rate limits.
- **Multi-path** (when permitted): test parallel egress routes; prefer lowest latency/packet loss.
- **VPN switcher** (policy-controlled): rotate gateways; geo-affinity only when legal/compliant.
- **Throttle-aware**: detect 429/5xx patterns; downshift concurrency; pre-cache assets.

B) Stealth Mode (power + discoverability)

- **Ultra-low duty cycle**: wake → sense → transmit PoA heartbeat → sleep (seconds to minutes).
- **Beacon minimalism**: encrypted, size-capped telemetry with onion-routed relay (when allowed).
- **Sensor triage** (on supported devices): prioritize GNSS / cell triangulation / Wi-Fi RSSI; loss-tolerant fusion.
- **Last-mile rescue**: if target is a child device, switch to **guardian policy**: location-only, anti-exfiltration, and audible/visual cues **only on command**.
- **Tamper-evidence**: even in stealth, PoA chain persists locally and syncs when safe.

Note: all privacy/child-safety features must respect local law and guardian consent. Stealth is opt-in, per-policy, and disabled by default.

C) Resource IQ

- **Power budget**: skills declare `timeLimitSec` & perf hints. The runtime schedules bursty work near charge events; pauses heavy tasks below battery thresholds.
- **Backpressure**: when the host is congested, the agent degrades gracefully—no stalls for the user.

Investor-grade moat (what wins the market now)

- **Proof-of-Action (PoA) cryptographic ledger**: every action is signed, chained, and exportable; evidences safety & ROI. (Verifiable-AI style ideas exist, but this specific PoA+tiers+dual-control stack remains rare in the wild. Related concepts are being discussed in industry blogs and research, but not as a turnkey agent OS.) cheqdMediumCoinCentral
- **Capability Passport**: a portable, signed artifact of tier unlocks, eval scores, and safety record—share with customers/investors.
- **Autonomy Bond**: optional insurance escrow tied to policy violations to align incentives and differentiate enterprise readiness.
- **Reproducible runs**: deterministic fixtures for audits—press “Re-run” and you get the same outcome + PoA.

Compared to “Holo1/Surfer” style web grounding (policy/localizer/validator), RogueOS pairs **that capability** with **verifiability, governance, dual-control, and reproducibility**—the set investors and enterprises are actually asking for in 2025. hcompany.aiDeloitte

---

Quick prior-art pulse check (you asked if you should enable research mode)

I did a **light scan just now**:

- HCompany surfaces Surfer-H + Holo1's policy/localizer/validator and web UI grounding; nothing publicly resembling your **PoA ledger + Capability Passport + Autonomy Bond** bundle as a productized stack. hcompany.aiHugging Face

- "Verifiable AI for agents" is being discussed (ZK proofs, verifiable credentials), but not in the concrete, ops-ready way you're packaging. cheqdMediumCoinCentral

This isn't a legal novelty opinion—it's a directional check. If you want a **broader prior-art sweep**, say the word and I'll assemble a citation pack you can hand to counsel.

What you can paste into blink.new (add this block to the earlier binding spec)

Addendum (SDK & Stealth Requirements — binding):

- Implement the **Skill SDK** exactly as specified (manifest + TypeScript interfaces).

- Enforce **Policy Gate** using `requiredTier`, `riskBudget`, and `dualControl`; fail-safe to lower tier on anomalies.

- Ship the **Web Form Submitter** example skill and its **deterministic fixtures**.

- Implement **Adaptive Network Layer** with profiles and throttle-aware behavior; add **Stealth Mode** as defined.

- Produce **PoA ledger** endpoints `/api/ledger/*` and an **exporter** that emits a zipped JSONL + artifacts package.

- Generate a **Capability Passport** (signed JSON) after the eval harness passes.

- Include an **Investor Mode** panel surfacing KPIs (success %, overrides, safety score, PoA latency, ROI calc).

---

want me to turn this into a **single PDF spec** (SDK + stealth + investor moat) you can drop into a deck or share privately? I can generate it now.

SHA-256 Hash for Verification:
bd0a29ea2d75c90cf60c2958ada2a824633e3f048869a
c87cbda38680f1aa554